Softwareparadigmen SS 2015, Übungsblatt 3

Abgabe: 10. Juni 2015, bis 16:00 Uhr vor dem Sekretariat IST, Infeldgasse 16b, 2. OG

Beispiel 1 (3.5P.)

Gegeben sei folgende syntaktische Erweiterung der Sprach AL $\mathcal{A}$ über dem Datentyp Integer $\mathbb{N}_0$ und Listen $\mathbb{L}$:
Wenn $v \in IVS$, $t_1 \in \mathcal{T}_{\mathbb{L}}$ und $a \in \mathcal{A}$, dann ist `foreach` v `in` t_1 `do` $a \in \mathcal{A}$
Definieren Sie eine formale Semantik für dieses Statement, erweitern Sie also die Interpretationsfunktion $I_{\mathcal{A}}$. Informell beschrieben, soll die Schleife über alle Elemente der Auswertung von t_1 iterieren. v nimmt dabei nach einander jedes Element aus t_1 an. Es darf angenommen werden, dass sich das Ergebnis der Auswertung von t_1 während der Interpretation des Schleifenkörpers nicht ändert.

Beispiel: Demonstrieren Sie die Funktionsweise ihrer Semantik-Definition indem Sie das folgende AL-Programm interpretieren:

```
begin
  z := 1;
  foreach i in [1,2,3] do
    z := mul(z,i);
end
```

Beispiel 2 (3P.)

Gegeben sei der Datentyp der *rationalen Zahlen*, der folgendermaßen definiert ist:

- Datentyp: $Q = (A, f_1, f_2, p_1)$
- Wertebereich: $A = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$, Paare (n, d) aus ganzen Zahlen, wobei der Nenner nicht Null sein darf
- Funktionen:
  1. f_1: add : $add((a, b), (c, d)) = (a * d + b * c, b * d)$
     Symbol auf syntaktischer Ebene: $\underline{add}$
  2. f_2: mult : $mult((a, b), (c, d)) = (a * c, b * d)$
     Symbol auf syntaktischer Ebene: $\underline{mult}$
- Prädikate:
  1. p_1: equal : $eq?((a, b), (c, d)) \leftrightarrow \frac{a}{b} = \frac{c}{d}$
  2. Symbol auf syntaktischer Ebene: $\underline{eq?}$
- Konstanten: zur Vereinfachung des Beispiels nicht vorhanden, für die Lösung können auf semantischer Ebene alle rationalen Zahlen verwendet werden

Bestimmen Sie für die nachfolgend angegebene Codierung der rationalen Zahlen in den Datentyp der *Listen + Integer*, für welche codierten Funktionen/Prädikate die Codierung gültig ist. Beweisen Sie für gültige Codierungen, dass die Codierungseigenschaften erfüllt sind, beziehungsweise geben Sie für nicht gültige Codierungen ein Beispiel an, dass die Codierungseigenschaften verletzt.

$$
\begin{aligned}
&\text{Codierung für Elemente aus } A\text{:}\\
\pi((n,d)) &= build(n, build(d, [])) = [n, d]\\
&\text{Codierung der Funktionen:}\\
\pi[add](x,y) &= build(first(x) + first(y), build(first(rest(x)) + first(rest(y)), []))\\
\pi[mult](x,y) &= build(first(x) * first(y), build(first(rest(x)) * first(rest(y)), []))\\
&\text{Codierung des Prädikats:}\\
\pi[eq](x,y) &= eq?(x,y)
\end{aligned}
$$

Beispiel 3 (3*P.*)

Gegeben sei der Datentyp *IntegerSet*, mit dem es möglich ist, Mengen von Ganzzahlen zu speichern. Definiert ist dieser Datentyp wie folgt:

- Datentyp: $IntegerSet = (A, f_1, f_2, p_1, p_2, c_1)$,
- Notation auf semantischer Ebene: Zur Unterscheidung von Listen verwenden wir zur Darstellung von Mengen geschwungene Klammern statt eckigen Klammern. Die leere Mengen soll durch $\{\}$ und eine Mengen mit n Elementen durch $\{i_1, \dots, i_n\}$ dargestellt werden.
- Wertebereich A: Der Wertebereich kann induktiv definiert werden: $A = \{\{\}\} \cup \{insert(s,i) | s \in A, i \in \mathbb{Z}\}$ Die leere Mengen ist also eine Mengen und jede Mengen kann mittels der *insert*-Funktion aus anderen Mengen erzeugt werden.
- Funktionen:
  1. f_1: $insert(s,i) = s'$ mit $s, s' \in A$ und $i \in \mathbb{Z}$
     - Fügt ein Element in die Menge ein
     - $insert(\{\}, i) = \{i\}$
     - $insert(\{i_1, \dots, i_n\}, i_{n+1}) = \{i_1, \dots, i_n, i_{n+1}\}$
     - $insert(\{i_1, \dots, i, \dots i_n\}, i) = \{i_1, \dots, i, \dots i_n\}$
     - Symbol auf syntaktischer Ebene: $\underline{insert}$
  2. f_2: $remove(s,i) = s'$ mit $s, s' \in A$
     - Löscht das nächste Element
     - $remove(\{\}, i) = \{\}$
     - $remove(\{i_1, \dots, i_k, \dots i_n\}, i_k) = \{i_1 \dots i_{k-1}, i_{k+1}, \dots, i_n\}$
     - $remove(\{i_1, \dots, i_n\}, i_{n+1}) = \{i_1, \dots, i_n\}$
     - Symbol auf syntaktischer Ebene: $\underline{remove}$
- Prädikate:
  1. p_1: $isEmpty?(s) = b$ mit $q \in A$ und $b \in \{T, F\}$
     - Ist wahr, wenn die leere Menge übergeben wird
     - Wenn $s = \{\}$ dann $isEmpty?(s) = T$
     - Sonst $isEmpty?(s) = F$
     - Symbol auf syntaktischer Ebene: $\underline{isEmpty?}$
  2. p_2: $isElement?(s,i) = b$ mit $q \in A$ und $b \in \{T, F\}$
     - Ist wahr, wenn i in s enthalten ist.

   - Wenn $s = \{\ldots i \ldots\}$ dann $isElement?(s, i) = T$
   - Sonst $isElement?(s) = F$
   - Symbol auf syntaktischer Ebene: $\underline{isElement?}$

- Konstanten:

  1. c_1: $\{\}$ für die leere Menge + Konstanten aus dem Datentyp Integer
  2. Symbol auf syntaktischer Ebene: $\underline{emptyS}$

Geben Sie eine Codierung des Datentyps *IntegerSet* in den Datentyp *Listen + Integer* an.

Beispiel 4 (3*P.*)

Interpretieren Sie die gegebenen prädikatenlogischen Ausdrücke über dem angegebenen Datentyp und bestimmen Sie deren semantischen Status. Geben Sie alle Zwischenschritte an und beachten Sie, dass für die Operatoren folgende Ordnung der Bindungsstärke gilt:

- $\neg$
- $\wedge$
- $\vee$
- $\rightarrow$

$\underline{(\forall x)(\exists y)\ eq?(mult(x, y), x)}$	...Datentyp $\mathbb{Z}$
$\underline{(\forall x) eq?(build(x, nil), nil) \vee eq?(1, x)}$	...Datentyp L + $\mathbb{N}$
$\underline{(\forall x)(\neg eq?(x, y) \ \rightarrow \ gt?(x, y))}$	...Datentyp $\mathbb{N}$